Cooperation Hidden within Competition in MARL

Stephen Mamosian

December 2025

Abstract

Competitive multi-agent environments often give rise to incentives that extend beyond purely adversarial behavior. While agents are typically trained to maximize individual reward, interactions among agents can reshape the effective objectives they face over time. Understanding how such interaction-driven dynamics emerge remains an important challenge in multi-agent reinforcement learning.

In this work, we investigate these dynamics using a competitive multi-agent variant of Tetris inspired by *Tetris-99*. Agents control independent boards, accumulate score by clearing lines, and influence opponents through a targeted sabotage mechanism. Although only one agent ultimately survives, the scoring objective is non-zero-sum, as agents benefit from prolonged gameplay that enables continued reward accumulation. We train agents through self-play and examine both learning progress and in-episode score trajectories. Our results show that agents learn effective gameplay strategies and exhibit nontrivial interaction patterns during the multi-agent phase, including delayed eliminations and shifts in relative performance. These findings suggest that strategic restraint can emerge in competitive settings as a consequence of the underlying incentive structure.

1 Introduction

Consider a competitive multi-agent game in which several autonomous players simultaneously attempt to maximize their own performance while operating within a shared, evolving environment. Although each agent interacts only with its own local game board, the agents are not isolated, and their actions can directly influence the difficulty or stability of others' environments. The resulting system exhibits a rich interplay between self-interested maximization and emergent inter-agent strategy, where short-term adversarial moves may conflict with longer-term collective benefit.

We model this setting through a multi-agent variant of Tetris inspired by the multiplayer game *Tetris-99*. In this environment, each agent controls an independent board and accumulates score by clearing lines. In the standard

Tetris-99 format, clearing lines automatically sends obstructing blocks to randomly selected opponents. Our environment retains the core idea that agents can affect one another's boards, but differs by granting agents explicit control over which opponent receives an attack. This targeted interaction mechanism introduces a direct channel through which agents can influence the competitive landscape.

The structure of the game creates an inherent tension. Although the termination condition is zero-sum—only one agent ultimately survives—the scoring objective is non-zero-sum. Agents seek to maximize cumulative score, yet opportunities to score diminish as opponents are eliminated. Competitive behavior may therefore include targeting strong opponents to reduce future competition or accelerating the elimination of weaker players to hasten game termination. At the same time, more subtle interaction patterns may emerge. In particular, agents may learn to prolong the multi-agent phase of the game by preferentially targeting stronger opponents while allowing weaker agents to remain active, thereby sustaining a larger population of competitors and enabling continued score accumulation.

Similar tensions arise in real-world competitive settings. In economic markets, a dominant firm may possess the resources to eliminate competitors through aggressive pricing or acquisitions, yet complete monopolization can be strategically suboptimal. Maintaining a competitive ecosystem and even occasionally supporting weaker competitors can expand total market value and sustain long-term profitability. More trivially, house cats may leave some mice alive to ensure their continued stay in a household. Motivated by these analogies, this project seeks to explore whether and when agents trained through self-play exhibit counter-intuitive interaction strategies in a competitive environment.

2 Methodology

2.1 Problem Setting

We model the multi-player Tetris environment as a stochastic Markov game with a finite set of agents $\mathcal{A} = \{1, \ldots, N\}$. Each agent controls an independent Tetris board and interacts with the environment in discrete time steps. The game proceeds synchronously, with all active agents selecting actions at each time step until a terminal condition is reached.

The environment admits a well-defined global state s_t at time t, consisting of the board configurations of all active agents, the currently falling piece and upcoming piece queue for each agent, and an indicator of which agents remain active. Given the current state s_t and the joint action $(a_1^t, \ldots, a_N^t)$ of all agents, the distribution over the next state s_{t+1} is independent of the interaction history, satisfying the Markov property.

At each time step, all agents observe the full global state. In particular, each agent has complete visibility of the board configurations and active pieces of all remaining agents. Consequently, the environment constitutes a fully observable

stochastic Markov game.

Boards evolve according to standard Tetris dynamics. At each step, an agent places its currently falling piece, potentially clears completed rows, and receives a new piece sampled from a fixed distribution. If blocks reach the top of an agent's board, that agent is eliminated and removed from further play.

2.2 Inter-Agent Attack Mechanism

Agents interact indirectly through an attack mechanism that serves as the primary point of strategic coupling in the environment. When an agent clears one or more rows on its board, it becomes eligible to send an additional Tetris piece to another active agent. Unlike the standard multiplayer format of *Tetris-99*, where attacks are assigned automatically, the proposed environment grants the sending agent explicit control over which opponent receives the attack.

Upon triggering an attack, the sending agent selects a target from the set of remaining active agents. A fixed sabotage piece is then inserted at the top of the recipient's board and must be placed immediately, directly influencing the recipient's board configuration and subsequent decisions. Although the shape of the sent piece is fixed, its impact depends on the current state of the recipient's board and can increase congestion or constrain future placements.

The game terminates when only a single agent remains active. While this survival condition induces a zero-sum outcome with respect to elimination, the reward structure of the environment is non-zero-sum. Agents accumulate cumulative reward proportional to the number of rows cleared over time, and opportunities to accrue additional reward diminish as opponents are eliminated. As a result, agents face a trade-off between aggressively reducing the number of competitors and maintaining a sufficient number of active agents to enable continued reward accumulation during the multi-agent phase of the game.

2.3 Single-Agent Training

Before training agents in the full multi-agent environment, we first train a single-agent model to play Tetris in isolation. This stage establishes a baseline level of competence in piece placement and board management, and it reduces the complexity of the subsequent multi-agent learning problem by decoupling low-level gameplay from strategic inter-agent decision-making.

In the single-agent setting, an agent interacts with an environment identical to the multi-agent formulation described above, except that no inter-agent attacks occur. The agent controls a single board, receives pieces drawn from the same distribution, and accumulates reward based solely on the number of rows cleared over time. Episodes terminate when the agent's board reaches the top.

The learning objective in this setting is to maximize cumulative reward through effective placement decisions that balance immediate line clears with long-term board stability. The resulting policy provides a foundation for competent gameplay, which is later transferred to the multi-agent setting, where

agents must additionally reason about opponent interactions and strategic piece sending.

2.4 Multi-Agent Training

Following single-agent pretraining, agents are trained in the full multi-agent environment described in Section 2.1. In this setting, multiple agents act concurrently, each controlling an independent board while interacting through the attack mechanism. Training is conducted using self-play, in which all agents are learning simultaneously and adapt to the evolving behaviors of their opponents.

Each agent acts independently based on its observation of the global state and maintains its own policy parameters throughout training. Agents jointly select actions at each time step, and the environment transitions according to the rules described in the problem setting. Rewards are assigned individually based on each agent's line clears and survival, and agents that are eliminated cease to receive further rewards.

From the perspective of any single agent, the learning environment is non-stationary, as state transitions and rewards depend on the concurrently evolving policies of other agents. To mitigate instability arising from this non-stationarity, agents are initialized with weights obtained from single-agent training and trained exclusively through self-play. This initialization provides a stable foundation for low-level gameplay, allowing agents to focus on adapting to strategic interactions in the multi-agent setting.

2.5 Agent Architecture

Each agent is controlled by a modular policy composed of two learned components. The first component selects placement actions for the currently falling Tetris piece based on the observed game state. The second component determines inter-agent attack decisions, including the target agent to be sabotaged.

The attack component conditions on the placement action selected by the placement component as well as the board configurations of all other active agents. Information produced by the placement component is transformed into a fixed intermediate representation before being provided as input to the attack component. This structure allows the attack component to reason about strategic interactions while remaining decoupled from low-level placement details.

Separating placement and attack decisions simplifies the learning problem faced by each component. The placement component focuses on effective board management, while the attack component learns how to influence opponents in response to gameplay outcomes. This modular organization improves training stability and enables controlled experimentation with different interaction strategies.

Both components are implemented as feedforward fully connected neural networks that output a scalar state-value estimate. The placement network takes an input vector of dimension `state_size` and consists of three hidden layers with 32 units each and ReLU activations, followed by a linear output

layer. The sabotage network takes an input vector of dimension `state_size_2` and consists of three hidden layers with 160 units each and ReLU activations, followed by a linear output layer.

Because the interface between the two components is fixed, the overall policy does not jointly optimize placement and attack decisions in an end-to-end manner. Instead, the components are trained concurrently through self-play within the shared environment. Exploring fully joint optimization within a single end-to-end model is left as future work.

2.6 Learning Algorithm and Loss Function

We train both the placement and sabotage components using a value-based reinforcement learning procedure with experience replay and one-step temporal-difference (TD) updates. Each component learns a value function over board states and selects actions by evaluating candidate successor states generated by the environment.

Placement decision. At each time step, the environment enumerates the set of legal placement outcomes, producing a set of candidate next board states. The placement network assigns a scalar value $V_{\theta_1}(s)$ to each candidate state, and the agent selects the placement whose resulting state attains the highest predicted value. Executing this placement yields an immediate reward r_t, proportional to the number of rows cleared, and a successor state s_{t+1}.

Sabotage decision. When an agent clears one or more rows, it triggers an opportunity to sabotage an opponent. The sabotage component evaluates candidate outcomes under full observability by conditioning on the current agent's board state and the potential next board states of each opponent. For each candidate opponent, the agent estimates a value for the resulting full-vision state and selects the sabotage target that maximizes the predicted value. Rewards associated with sabotage decisions are defined as the cumulative reward accrued since the previous sabotage event.

TD targets and loss. For each component, transitions are stored in a replay buffer and minibatches are sampled uniformly during training. Each stored transition consists of a current state s_t, the resulting successor state s_{t+1}, the immediate reward r_t, and a termination indicator $d_t \in \{0, 1\}$. Learning is performed using one-step temporal-difference (TD) updates, which adjust the estimated value of a state based on the reward observed and the model's current estimate of the value of the next state.

Given a sampled transition (s_t, s_{t+1}, r_t, d_t), the temporal-difference target is defined as

$$y_t = r_t + (1 - d_t)\,\gamma\, V_\theta(s_{t+1}).$$

The discount factor $\gamma \in (0, 1)$ controls the relative importance of future rewards, with larger values encouraging long-term strategies that prioritize sus-

tained gameplay over immediate gains. When the episode terminates ($d_t = 1$), no future value is available and the target reduces to the immediate reward. Otherwise, the target incorporates the discounted value estimate of the successor state, enabling incremental learning without waiting until the end of an episode.

Network parameters are updated by minimizing a squared regression loss of the form

$$\mathcal{L}(\theta) = E\left[\left(V_\theta(s_t) - y_t\right)^2\right],$$

which penalizes discrepancies between the network's current value prediction and the TD target. Training is performed using experience replay with ϵ-greedy exploration, and both components are updated at the end of each episode using minibatches sampled from their respective replay buffers.

3 Experimental Results

3.1 Experimental Setup

Experiments are conducted in the multi-agent Tetris environment described in Section 2, with five agents acting concurrently. Each agent controls an independent board and interacts with other agents through the sabotage mechanism. Episodes proceed until only a single agent remains active. To focus on interaction dynamics that arise specifically in the multi-agent setting, our analysis restricts attention to timesteps in which at least three agents remain alive. The subsequent two-agent endgame is excluded from analysis, as the multi-agent strategic targeting dynamics are no longer present in this phase.

3.2 Quantitative Results

Figure 1 shows agent average scores over the course of training episodes. During the initial phase of training (approximately the first 600 episodes), performance remains largely static and low. Beyond this point, the height of the observed performance peaks increases substantially and more frequently, although low-scoring episodes continue to occur. This indicates that high-performing behaviors become achievable in later training stages, despite the continued variability in episode-level outcomes. As a whole, these trends show that the agent learns to play the game in a way that yields higher scores over the course of training.

Figure 2 shows agent score trajectories within a single multi-agent episode in which each agent uses a distinct trained model. Agent performance separates early in the episode, with noticeable differences in cumulative score emerging among agents. By approximately step 100, several agents have been eliminated. Player 0 maintains a substantially lower score than the others while remaining active longer than some higher-scoring agents, whereas the remaining agents accumulate score at different rates prior to elimination.

Figure 3 shows agent score trajectories within a single multi-agent episode in which all agents use identical trained model parameters. In this setting,

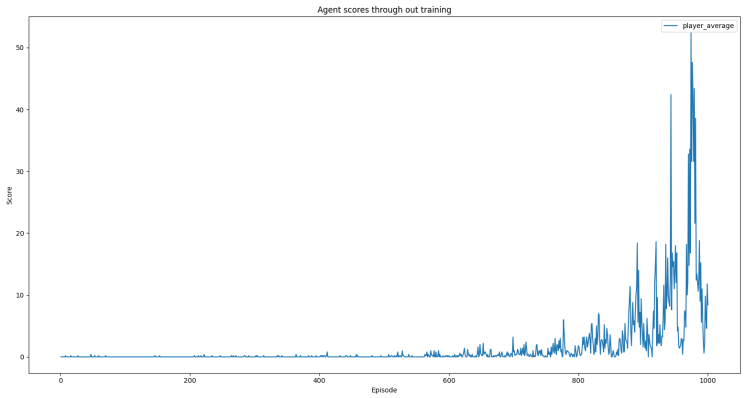

Figure 1: Agent scores over training episodes

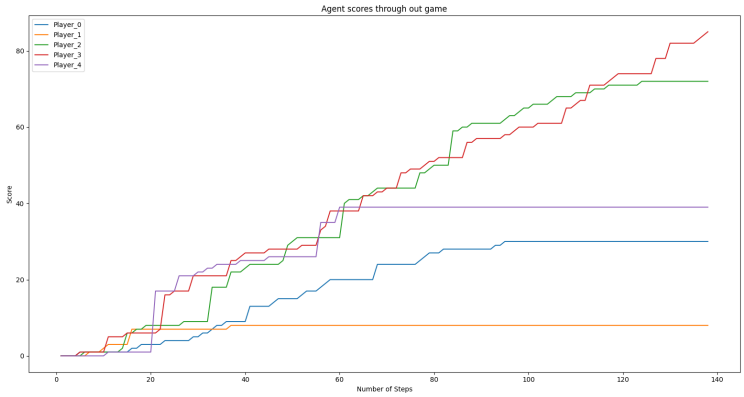

Figure 2: Agent scores within a single multi-agent game with distinct trained models for each agent

agent scores remain closely clustered during the early and mid stages of the episode, and four agents remain active until approximately step 160. Relative performance differences emerge more gradually than in the heterogeneous-model case. Although one agent quickly establishes and keeps lead, the scores across the other agents do not fall noticeably behind.

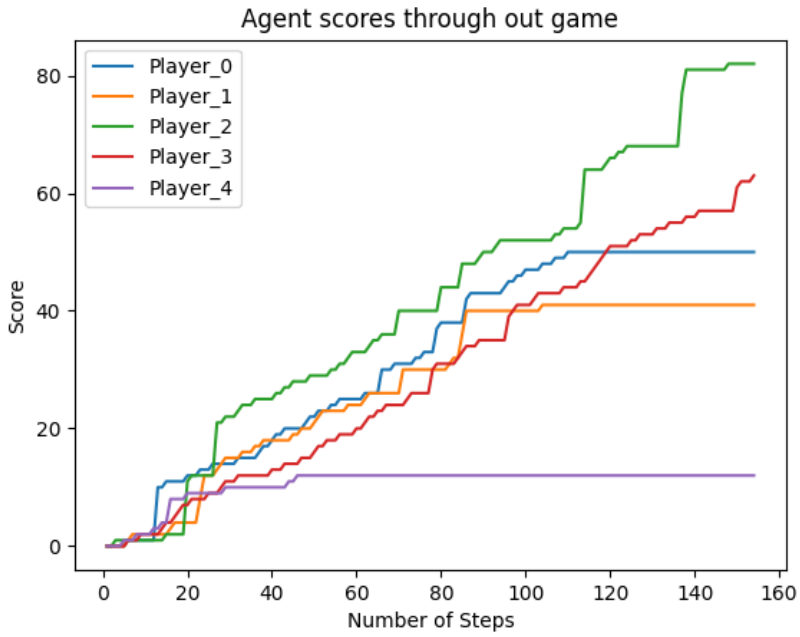

Figure 3: Agent scores within a single multi-agent game with a shared training model across all agents

4 Discussion

The results presented in Section 6 highlight interaction dynamics that are not visible from training curves alone. While the training graph shows that agents learn increasingly effective Tetris strategies over time, the in-episode score trajectories provide a clearer picture of how agents behave when competing simultaneously in a shared environment.

In the heterogeneous-model setting shown in Figure 2, agent performance separates early in the episode, with noticeable differences in cumulative score emerging among agents. Despite this divergence, one agent remains active for a relatively long duration while maintaining a substantially lower score than several higher-scoring opponents. This suggests that elimination timing is not determined solely by short-term performance. One plausible explanation is that agents tend to focus their attacks on stronger opponents, effectively deprioritizing weaker agents who pose less immediate threat. This behavior is consistent with a form of strategic restraint in a multi-agent setting, although other factors—such as stochastic gameplay effects or board-specific dynamics—may also contribute.

When all agents use identical model parameters, as shown in Figure 3, a dif-

ferent pattern emerges. Score trajectories remain closely clustered for a longer portion of the episode, and eliminations are delayed. Multiple agents survive well into the game with similar cumulative scores, and relative advantages develop more gradually. This suggests that when agents have comparable underlying competence, interaction effects play a larger role in shaping gameplay dynamics than raw skill differences alone.

Taken together, these observations suggest that agents may learn to condition their behavior on the relative strength of opponents rather than simply attempting to eliminate any available target as quickly as possible. The persistence of lower-scoring agents and the delayed emergence of a dominant leader are consistent with policies that implicitly balance short-term competitive pressure against longer-term scoring opportunities. At the same time, the evidence presented here is qualitative and based on individual episodes, and thus cannot conclusively establish the presence of cooperative intent.

Finally, it is important to emphasize that these interaction patterns are specific to the phase of the game in which three or more agents remain active. Once gameplay transitions to a two-agent end state, the strategic landscape changes substantially, and interactions reduce to direct competition. As a result, any conclusions regarding cooperative or balancing behavior should be understood as applying only to the multi-agent portion of the game.

5 Limitations and Future Work

While the results presented in this work provide insight into multi-agent interaction dynamics in a competitive Tetris environment, several limitations of the current implementation suggest natural directions for future exploration.

First, agents are trained for a finite number of episodes, and learning dynamics remain noisy even in later stages of training. Although high-performing behaviors emerge over time, extended training may allow policies to stabilize further and enable clearer separation between transient behaviors and persistent interaction strategies. Longer training horizons could also help determine whether the observed patterns generalize across episodes rather than arising from individual trajectories.

Second, the agent policy is implemented as a modular architecture composed of separate placement and attack components with a fixed interface. While this design simplifies training and improves stability, it prevents fully joint optimization of placement and sabotage decisions. As a result, placement actions are not explicitly optimized with respect to their downstream strategic impact on inter-agent interactions. Future work should look to replace this modular design with a single end-to-end model that jointly reasons about placement and interaction decisions within a unified policy.

Third, in the current environment, agents select only the recipient of a sabotage action, while the type of piece sent is fixed. This restricts the space of possible interaction strategies and precludes intentional assistance through beneficial piece selection. Allowing agents to choose both the target and the piece

type would enable a richer set of behaviors, including more explicit forms of cooperation or alliance formation. However, this extension substantially increases the action space and would likely require additional architectural and training modifications to remain tractable.

Additionally, the experimental analysis focuses on qualitative behavior observed in individual evaluation episodes. While these episodes reveal informative patterns, broader statistical evaluation across many runs would strengthen conclusions regarding the prevalence and robustness of the observed dynamics. Future work could incorporate systematic aggregation over multiple episodes, alternative performance metrics, and controlled ablation studies to better isolate the effects of interaction mechanisms and policy design choices.

Finally, the current analysis assumes full observability of all agents' board states. Relaxing this assumption to partial observability would more closely resemble realistic competitive settings and introduce additional challenges related to inference and uncertainty. Investigating how limited information affects learned interaction strategies represents another promising avenue for future research.

6 Conclusion

In this work, we studied a competitive multi-agent variant of Tetris designed to expose tensions between individual performance maximization and interaction-driven dynamics. By allowing agents to directly influence one another through targeted sabotage, the environment creates a setting in which short-term competitive incentives can conflict with longer-term scoring objectives.

Our results show that agents learn increasingly effective Tetris gameplay strategies through self-play and that their in-episode performance trajectories exhibit nontrivial interaction patterns during the multi-agent phase of the game. In particular, score dynamics within individual episodes suggest that elimination timing is not solely determined by immediate performance, and that agents' behaviors are shaped by the presence and relative strength of other competitors. Comparisons between heterogeneous and shared-model settings further highlight the influence of base policy competence on multi-agent dynamics.

While the evidence presented here does not establish explicit cooperative intent, it demonstrates that learned policies can give rise to interaction patterns consistent with strategic restraint in competitive environments. These findings underscore the importance of studying agent behavior at the level of interaction dynamics rather than aggregate performance alone. Overall, this work contributes a controlled experimental setting for examining emergent behavior in multi-agent reinforcement learning and motivates further investigation into richer interaction mechanisms and end-to-end learning approaches.